*ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method* • Nashville, TN • April 19-23, 2015, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2015)

# SIMPLIFYING WORKFLOW FOR REACTOR ASSEMBLY AND FULL-CORE MODELING

**Rajeev Jain* and Vijay Mahadevan**
Argonne National Laboratory
9700 S. Cass Ave, Argonne, IL 60439
*jain@mcs.anl.gov, mahadevan@anl.gov,

**Robert O'Bara**
Kitware Inc., Albany, NY
bob.obara@kitware.com

## ABSTRACT

The evolution of efficient scalable solvers and multiphysics codes in the past decade has considerably increased the requirements on computational needs for high-fidelity simulations of nuclear reactors through resolution of more heterogeneity in the physical models. In this paper, we present the Reactor Geometry (and mesh) Generator (RGG) toolkit, which is a part of MeshKit library developed at Argonne National Laboratory. The extensions of the RGG toolkit have been used to create PWR, ABTR, VHTR, MONJU, and several other types of reactor geometry and mesh models for use in state-of-art physics solvers. RGG uses a lattice hierarchy-based approach to create these reactor core models and has been designed to scale on even large models with up to 1 billion hexahedral elements. Details on the RGG methodology and description of the parallel aspects of the tool are provided. Several model full-core reactor problems (PWR, ABTR) are also presented along with openly accessible mesh files for reproducibility. A GUI for RGG tools (AssyGen and CoreGen) is available currently in the open-source domain for all popular operating systems and can significantly simplify the complexity in mesh generation for nuclear reactors.

*Key Words*: Nuclear Reactor Core Mesh, Nuclear Reactor Core Geometry, RGG, MeshKit

## 1 INTRODUCTION

Accurate nuclear reactor design and modeling of safety accidents require well-resolved geometry models that are complex to describe traditionally. The creation of the computational meshes with good-quality elements adds several complications, especially when the geometries of the core, assemblies, and fuel pins are nonuniform. Typical nuclear reactor core designs are described by a lattice of cylindrical pins with a duct that serves as a reflector and shield. The pins can contain fuel material with varying enrichments, absorbing material for controlling the nuclear chain reaction, or instrumentation with several axial regions, thereby increasing the total number of materials to be defined exponentially. The design of reactors with a combination of these pins arranged in either a rectangular (PWR, BWR) or hexagonal (SFR, VHTR) lattice covers more than 80% of the designs that are modeled by reactor analysts. Hence efficient geometry model creation tools and mesh generators to compute well-conditioned elements, optimized for physics solvers are critical in the accurate analysis of these nuclear reactor problems.

*corresponding author

Generating meshes that accurately describe and resolve reactor core geometry is in general a time-consuming and labor-intensive process. Computational geometries and meshes describing a reactor core model can take several days to weeks to generate. Complicated structures with conical pins and small volumes adjacent to very large volumes tend to impose a significant set of requirements on the geometry and mesh generation. The overall geometry of a single reactor assembly may comprise several thousand geometric regions, when taking into consideration the materials and axially varying properties of the assembly. With a small change in the model, such as a change in the radii of a fuel pin or control rod in place of the coolant, the complete meshing process must be repeated. Specialized meshing tools have been developed in several fields of simulations; however, little or no effort has been invested in creating a specialized tool for generating reactor core geometry and meshes. RGG takes advantage of the repeated lattice-based structure and allows for a balance of user control and automation throughout the process. Various reactor types such as MONJU, EBR-II, 1/6 VHTR, and HTGR have been modeled by using RGG [1, 2].

In the subsequent sections, we provide details on the RGG methodology and introduce the typical workflow for modeling rectangular and hexagonal lattice cores. The computational meshes generated with the CoreGen and AssyGen tools for a PWR and an ABTR core are also discussed.

## 2 MESHKIT RGG APPROACH

MeshKit, an open-source mesh generation library, is part of SIGMA toolkit [3] from Argonne National Laboratory. MeshKit leverages the robust infrastructure provided by SIGMA libraries, namely, CGM (Common Geometry Module) [4] for geometry and MOAB (Mesh-Oriented DataBase) [5] for mesh data-structure. Several algorithms in MeshKit have been exposed through tools and applications that are also usable for general geometry description. Various geometry file conventions or formats such as OCC/ACIS/Parasolid/Native geometry are supported by CGM. Additionally, MeshKit can output the mesh in several file formats using the I/O interfaces provided by MOAB, such as the native HDF5 format, NetCDF (exodus), unstructured VTK, and several commercial formats (STARCCM+, ABAQUS, ANSYS).

The RGG methodology involves three primary steps. This overall workflow is highlighted in Fig. 1. AssyGen is the first step of the three-step core mesh creation process implemented in RGG. AssyGen reads an input file describing a reactor assembly lattice and generates an ACIS or OCC-based geometry file, along with a template script for generating a mesh for the assembly using the CUBIT meshing toolkit [6]. The second step is meshing, where the user may choose to perform meshing using the CUBIT mesh script generated by AssyGen or using native meshing algorithms or external interfaces to Netgen in MeshKit. AssyGen supports several features, such as axial numbering material and boundary conditions, sectioning, rotation, and information about location of pins.

In the third step, the CoreGen tool reads an input file describing the reactor core arrangement (lattice structure) and generates the reactor core mesh or geometry from its component assemblies. The CoreGen tool uses CopyMesh, ExtrudeMesh, CopyGeom, and MergeMesh algorithms in MeshKit. Figure 1 shows the two assembly meshes and an interstice mesh file that is used to create a 19-assembly reactor core model. A *makefile* is generated by CoreGen to automate this process for repeated modifications. Hundreds of assembly meshes

along with interstices meshes such as grid spacers and restraint rings form a complete reactor core mesh. These models can be large, involving up to several billion mesh elements. Generation of such models on a single computational node (serial) can hence be a time-consuming process. Parallel-enabled CoreGen has been used to create large meshes such as MONJU, VHTR, HTGR, EBRII, and PWR for different reactor simulation codes in order to considerably speed the mesh generation process. The scalability results for VHTR and MONJU reactor designs using the parallel version of CoreGen were published in [7]. In the parallel version, all processors read the CoreGen input file, parse, and determine assembly copies assigned to this processor based on a round-robin distribution or a distribution based on frequency of occurrence of each assembly in the core. Then, each processor locally reads assembly meshes and performs assembly copy/move operations assigned to this processor. After this stage, each processor performs a local merge, followed by a parallel merge of meshes between processors where the communication kernels exposed by MOAB are utilized. The material and boundary condition data are specified with COPY/EXPAND or EXTRUDE set (metadata) handling. Parallel HDF5 I/O provided by MOAB is then used to save the final mesh.

In this setup, if only one assembly or pin is changed in an assembly, RGG does not run the entire meshing problem again. Only the assembly that has been modified is remeshed, and the core model is created (merge mesh). Specifications of the edge interval, axial, and radial mesh sizes for all assemblies are provided. We note that the user must be careful that the meshes between assemblies and the instrumentation merge and form a conformal core mesh. More details and features of RGG tools are explained, along with examples, in Sec. 3.

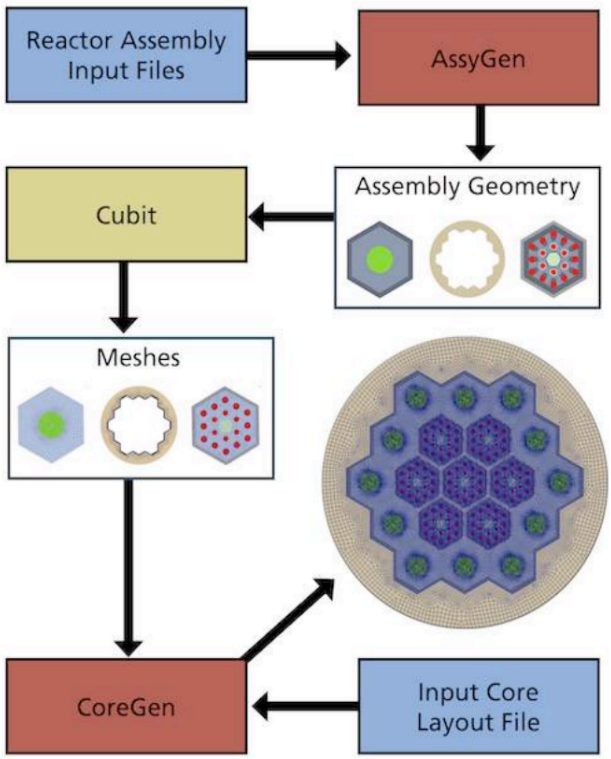



**Figure 1. RGG workflow.**

# 3 DESIGNING REACTOR ASSEMBLIES AND CORE

RGG can handle both rectangular and hexagonal reactor core models. In this section, we discuss some of the features of both types of cores, give a general introduction to modeling with RGG, and present several detailed core models.

To simplify the visualization and definition of the geometric surfaces comprising reactor models, Kitware Inc. [8], in collaboration with Argonne researchers, has developed an RGG GUI application. Five major steps are involved in defining a reactor geometry model using the RGG GUI tool and in general using the AssyGen and CoreGen tools of MeshKit.

1. *Designing Pins*: After the initial choice of rectangular/hexagonal core, the first step is to define the pins. mesh sizes, material ,and boundary conditions.
2. *Designing Ducts*: Once the pins are created the dimensions of ducts around the pins are specified. The preview of the model is available instantly upon creation.
3. *Designing Assemblies*: All the assemblies must be defined and steps 1 and 2 are repeated as required.
4. *Setting Up Core Layout*: A core layout from assemblies described in step 3 along with other core level parameters such as interstice meshes, boundary conditions are assigned.
5. *Generating Mesh*: This step invokes AssyGen, Meshing, and CoreGen to create the final model.

The RGG GUI comes with a detailed user manual that describes all these steps and various other useful options.

## 3.1 Simple Pin Cell Model

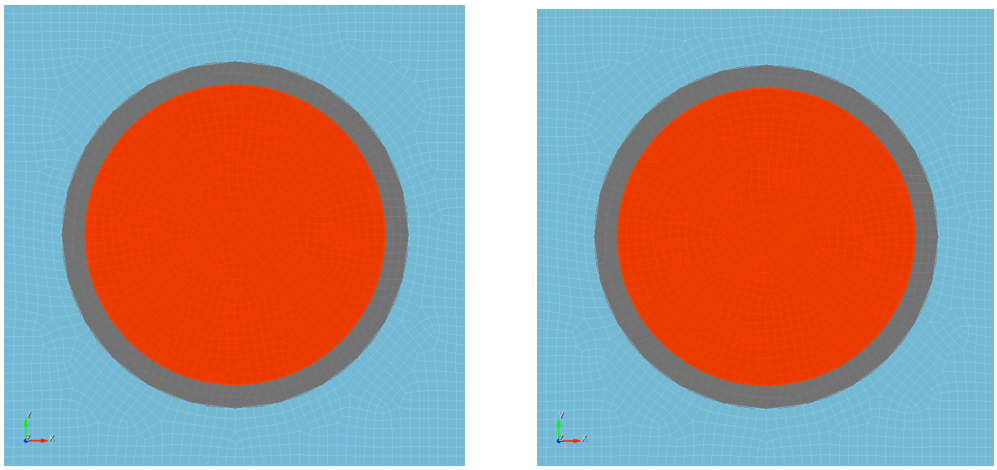


**Figure 2. Simple pin cell: (a) coarse mesh and (b) fine mesh.**

The RGG toolkit can be accessed by means of the command line interface and text-based input files (see blue boxes in Fig. 1) or via the GUI application. Both the command line interface and the GUI follow the approach shown in Fig. 1. The text-based RGG input files can also be

loaded into the GUI for visualization and further modifications. The meshes shown in Fig. 2 were developed through the definition of two circular, concentric pins with varying materials surrounded by a coolant material. The tools use CUBIT [6] in the background to generate the necessary meshes based on sizing functions specified by user.

### 3.2 Assembly Model

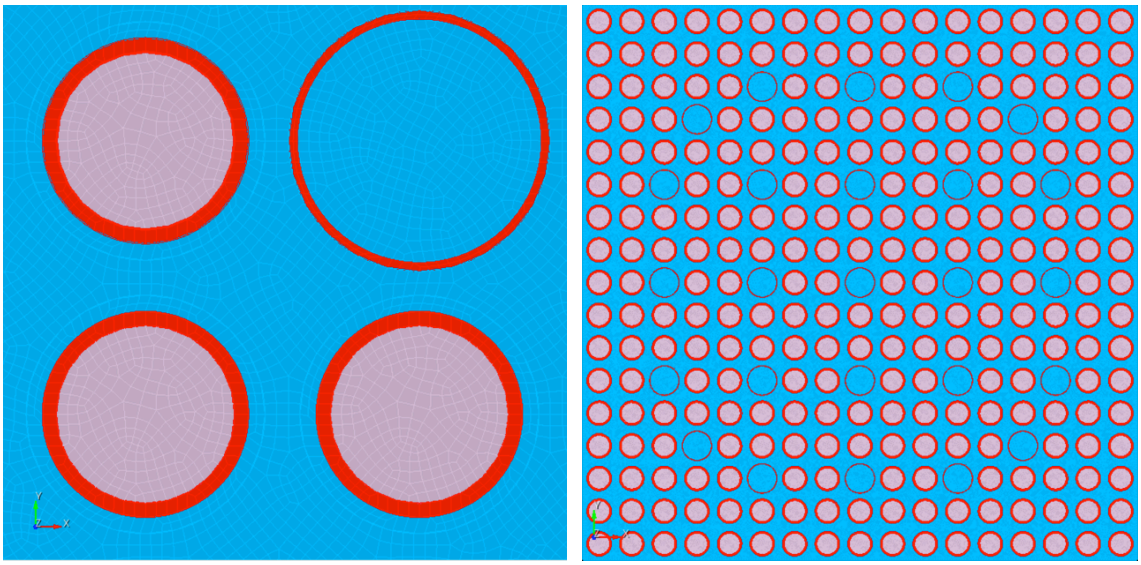


**Figure 3. Assembly models: (a) 2x2 assembly and (b) 17x17 assembly.**

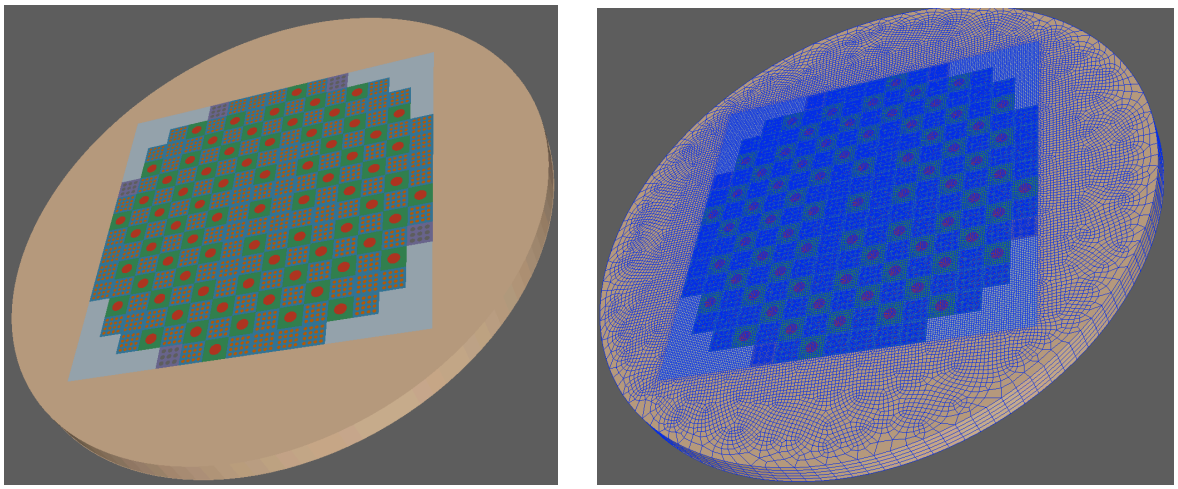
For modeling the 2x2 and 17x17 conical pin assembly models shown in Fig. 3, as well as the 8x8 model, AssyGen was used to create the geometry, material, and boundary condition details and to pass them to the CUBIT mesh script with appropriate mesh sizing. Mesh sizes were controlled by AssyGen keywords to specify radial mesh size and edge mesh interval. A scheme selection of "hole" must be used for meshing the concentric rings to describe the fuel material inside the clad.

The generation of a mesh may fail when using regular paving to mesh assemblies that contain several concentric pins. A pin that is larger than the pitch of the assembly can be created by using RGG by filling the area around that pin with empty pins (Fig. 3(a)); RGG automatically fills that space with background material (coolant). If the user does not specify sizing controls for the mesh, RGG can additionally compute and apply sizes automatically for meshing the reactor assemblies.

### 3.3 Rectangular Full-Core Models

The assembly and pin cell models in Secs. 3.1 and 3.2 are part of creating a full-core model. Since the arrangement of various assemblies forms a complete core model, detailed assembly descriptions can lead to very large models that cannot be handled within the memory constraints of available desktop computers. The parallel version of CoreGen has a substantially reduced memory footprint because of the parallel copy/move, merge, and save operations. Each individual assembly mesh is written as a separate mesh file, which serves as input to the CoreGen program to create the final core model. The final output created by the serial and

parallel versions of the CoreGen process results in the same final mesh with material and boundary conditions specified as desired by the analyst.



(a) (b)

**Figure. 4. Core models: (a) geometric model for 193-assembly reactor and (b) mesh model of Fig. 4(a). Height of the reactor has been scaled for better visualization control.**

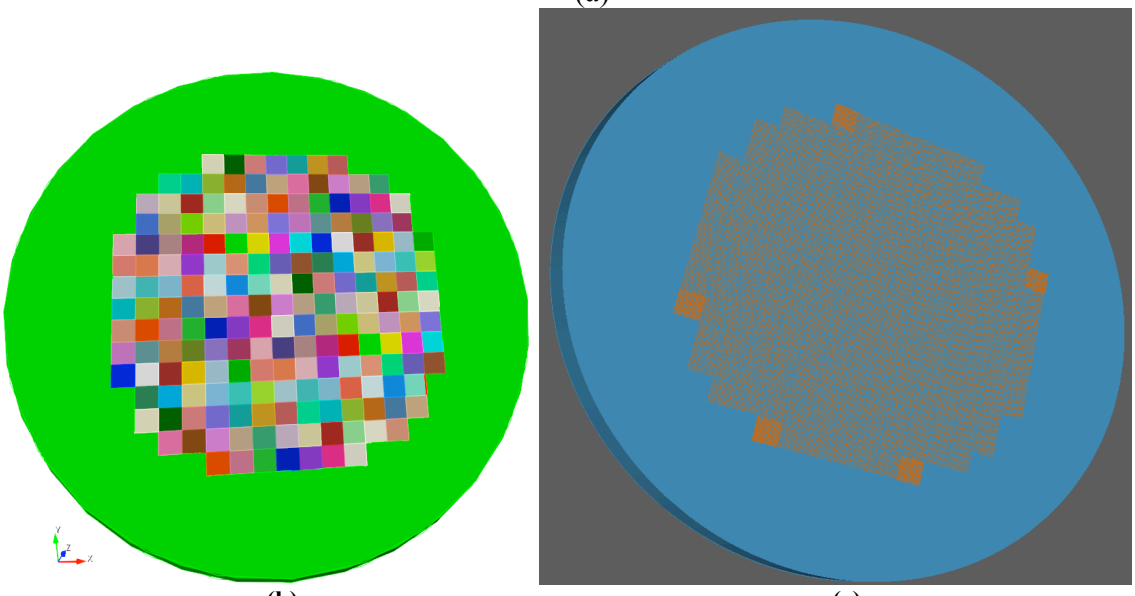
### 3.3.1 Four-Loop PWR Core

The Westinghouse pressurized water reactor nuclear power plant document available online [9] describes the reactor core model shown in Fig. 4(a). The core model consists of 193 total assemblies, and geometrically there are three different types of assemblies. Core model shown in Figs. 4(a) and 4(b) are formed with assemblies with fewer pincells (1x1, 3x3 and 4x4). In the actual model, each assembly is a 17x17 lattice (shown in Fig. 3(b)); the complete reactor core model shown in Fig. 5(c). The final mesh model is hard to visualize since the mesh size is too small for viewing, and hence only the geometry and a simplified model (Fig. 4) are shown here. The complete core creation for this model is automatic; that is, small changes in material, assembly, or pincell arrangement and the creation of the outer vessel are handled automatically by RGG. We note, however, that specialized geometries such as grid spacer creation are not supported yet and must be modeled separately. When the outer vessel or other external meshes are modeled, the surfaces and curves where the reactor assembly and external pieces meet must have coincident nodes for the merge process to work and the conformal core mesh model to be created. If these conditions are not met, the CoreGen process will fail with appropriate error messages.

On 192 CPU cores, the final mesh model for this core contains 6 million 3D hexagonal elements, which can be created in less than 10 minutes from scratch. A majority of this time is taken in the serial assembly and interstices mesh generation process. The parallel CoreGen step takes only 90 seconds to assemble the core and create the final mesh; a majority of the time is taken by the parallel merge and save steps. It must be noted that the file size of this mesh on disk is 1.1 GB and the maximum memory used by a processor during the mesh generation process for this model was about 620 MB. Memory results from this and other examples presented in this section show that the parallel CoreGen process considerably reduces the memory footprint of the overall mesh generation process.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| xx | xx | xx | xx | INST | Fuel | Control | Fuel | Fuel | Fuel | Fuel | xx | xx | xx | xx |
| xx | xx | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | xx | xx |
| xx | Fuel | Fuel | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | xx |
| xx | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | xx |
| Fuel | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | INST |
| Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel |
| Fuel | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | Fuel | Fuel | Fuel | Fuel | Control | Fuel | Control |
| Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel |
| Control | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | Fuel | Fuel | Control | Fuel | Control | Fuel | Fuel |
| Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel |
| INST | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | Control | Fuel | Control | Fuel | Fuel |
| xx | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | xx |
| xx | Fuel | Fuel | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Fuel | Fuel | xx |
| xx | xx | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | Control | Fuel | xx | xx |
| xx | xx | xx | xx | INST | Fuel | Fuel | Fuel | Control | Fuel | INST | xx | xx | xx | xx |

(a)



(b) (c)

**Figure 5. PWR core model: (a) 2D layout of the four-loop reactor core (editable using RGG GUI), (b) CoreGen geometric core creation is shown with homogenized assemblies, and (c) geometric model with 193 hetrogenous assemblies and outer core vessel (assemblies have 17x17 pins).**

.

### 3.3.2 1/4 PWR Core

Figure 6 shows the benchmark problem "MOX Fuel Loaded Small PWR Core"; a detailed description can be found on the website of the Nuclear Reactor Analysis and Particle Transport Lab [10]. Individual assembly geometries are created by using AssyGen, which is then utilized by CoreGen to copy/move the assemblies and form the core geometry. The model consists of approximately 11K volumes, 5.2M hexes, and 5.9M vertices. On a Linux desktop, the assembly geometry creation takes 8 minutes, and CoreGen takes 12 minutes of wall-clock time and uses 0.9 GB of RAM for creating the core geometry model from component assembly geometries (no mesh generation). The maximum number of processors that can be used by CoreGen for this problem is 25, which is the total number of assemblies in this core model. Using 25 processors, CoreGen takes 45 seconds and 210 MB maximum memory to create the mesh; the majority of the time is used in merging the nodes between the assemblies. The serial version of CoreGen takes 1.9 minutes and 1.7 GB of RAM. The copy/move takes 42 seconds, and the serial merge takes 50 seconds.

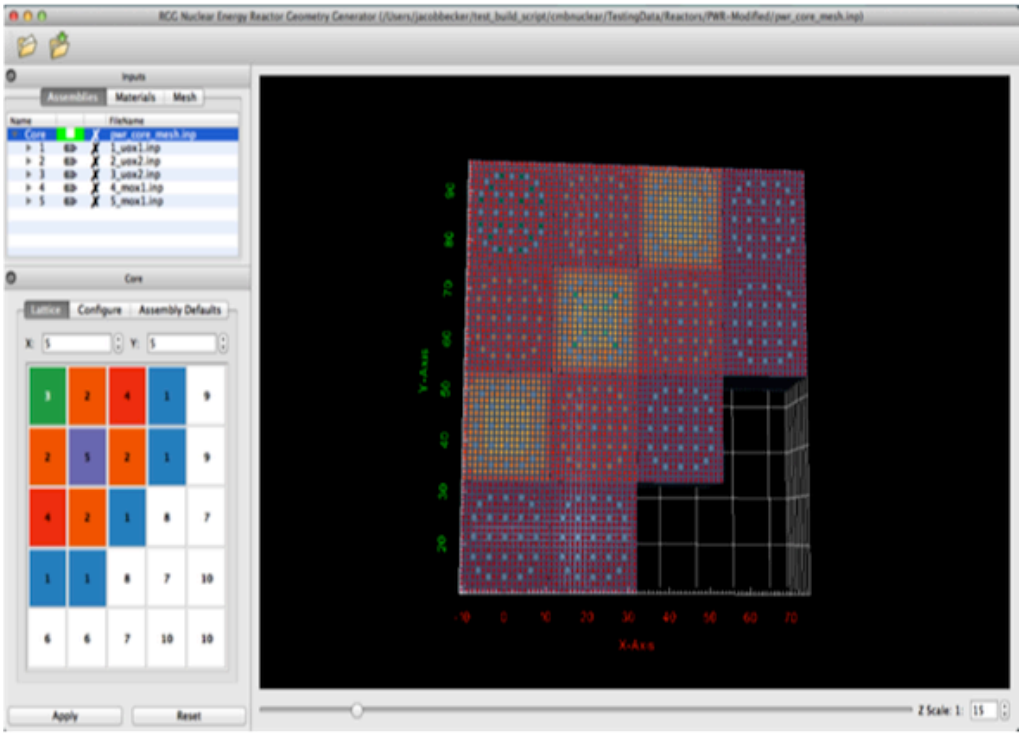



**Figure 6. PWR reactor core with five types of assemblies modeled using RGG.**

## 3.4 Hexagonal Full-Core Models

RGG tools have been used to generate several heterogeneous single assembly and full core meshes (SFR, VHTR, ABTR, etc.) to be used in high-fidelity physics codes to solve multiphysics problems accurately. Details on some specific hexagonal lattice cores are given here.

### 3.4.1 VHTR Full Core

**Figure 7. Full-core VHTR; red square region shows zoomed view from top to bottom.**

Figure 7 shows the geometric model of a full VHTR core modeled with the standard RGG process described above. The differences between the rectangular and hexagonal lattices are specified based on reactor types, controlled by the user during model creation. The geometric model contains 313 assemblies with 33K volumes, and it takes 5.5 GB of serial memory and 30 minutes to create on a standard Linux desktop workstation. RGG also supports creation of hexagonal reactor cores with 1/6 and 1/12 symmetry. When creating a fraction of hexagonal assemblies using RGG, sectioning and rotation of assemblies are used to form the core model. The 1/6 VHTR core model was described in our previous paper [7].

### 3.4.2 ABTR Full-Core Model

CoreGen is run to create the final homogenized core model from assembly meshes and an outer covering (duct) mesh specification.

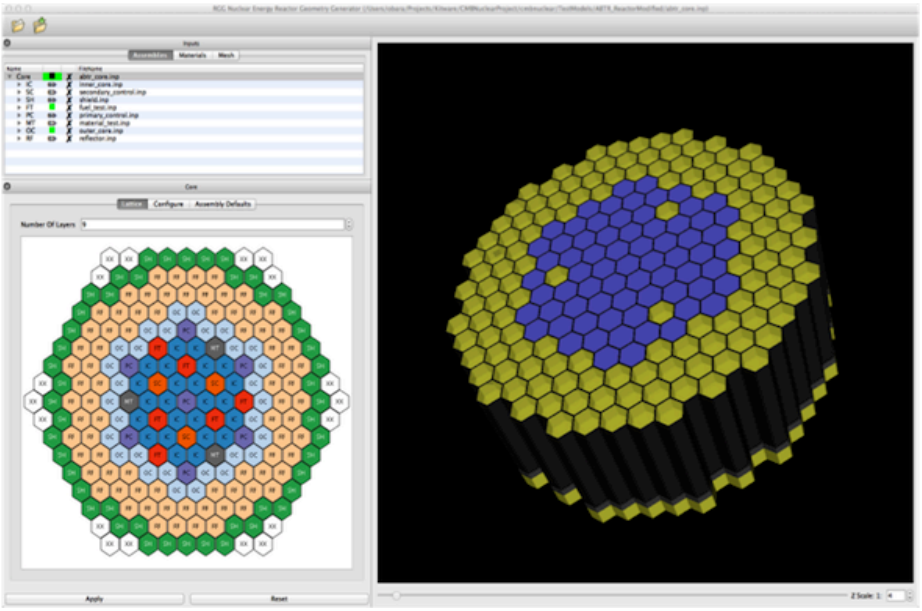


**Figure 8. ABTR full core.**

The coarse ABTR model shown in Fig. 8 consisted of 800K hex elements. The core mesh consists of about 7,200 material blocks to provide better spatial resolution for the physics solvers. CoreGen takes 5 seconds to assemble the entire core mesh using 128 processors in parallel. The maximum memory used by a single processor is only about 110 MB. The serial runtime for CoreGen is 45 seconds with 280 MB of memory used. For the finer mesh for the same ABTR core model each assembly consists of 5–10M hexahedral elements. The mesh size of the finer core model is 1.2B hexahedral elements and the resulting mesh file size is 137 GB. CoreGen on 200 processors (each processor loads one mesh file: 199 assembly + 1 interstices mesh file) takes 30 minutes, of which the parallel save takes up 90% of the time and the parallel merge takes 8%. The maximum memory used by a processor is 1.96 GB. This model is impossible to create serially on conventional CAD workstations.

## 4 CONCLUSION AND FUTURE WORK

MeshKit provides a robust parallel infrastructure for geometry/meshing researchers, tool developers, and users needing to generate geometry and meshes. MeshKit has most of the traditional meshing algorithms required for meshing complex geometries. The RGG tools AssyGen and CoreGen are part of MeshKit; both tools use text-based input files. These input files are based on a set of keywords that help define the geometry and meshing parameters for creating a nuclear reactor core. RGG GUI enables users to build the reactor model without writing the text-based input files explicitly. The GUI is intuitive and easy to understand; it comes with a users manual; and it works with Linux, Mac OSX, and Windows operating systems. The RGG GUI developed by Kitware as a part of an SBIR can display the geometry and meshes created by AssyGen and CoreGen, respectively.

MeshKit is ideally suited for development of new meshing tools and algorithms. Simulation of complex systems such as nuclear reactors requires detailed models that properly capture the geometric shape and have correct specification of material and boundary conditions. Different physics such as neutron transport, fluid flow, thermal expansion, and heat transfer must be studied in order to fully understand the performance and safety aspects of nuclear reactors. The parallel RGG tools enable the creation of such large and complicated reactor models for different physics simulations. Several enhancements and fixes to the AssyGen and CoreGen tools in RGG, including introduction of shifting material and Neumann set ids, have been incorporated in the current latest version. With the introduction of the new distribution scheme for parallel CoreGen, we can currently create meshes such as the finer ABTR core mesh (file on disk is 137 GB and consists of more than a 1 billion hexahedral elements). Such meshes are impossible to construct by using serial meshing processes on a standalone workstation.

We plan to create fully detailed models and use coupled physics solvers to analyze some interesting nuclear reactor safety problems with these models. Work is in progress for developing geometry partitioning-based meshing algorithms to mesh the assemblies in parallel. Also, new schemes are being formulated for better load balancing during copy/move task distribution. This scheme will be a combination of existing schemes with more weightage given to meshes with larger element count. Future efforts include parallel AssyGen development, which would enable creation of individual pins and components in parallel, resulting in substantial speedup of mesh generation on very high core counts. The ability to add spacer grids (for LWRs) and wire wraps (for SFRs) will also be implemented in order to better model the heterogeneity in the geometry for these reactors.

## ACKNOWLEDGMENTS

We thank the SIGMA group at Argonne for developing the libraries necessary for MeshKit and RGG to work efficiently. We also thank Kitware Inc. for developing the RGG GUI tool and for helping MeshKit become more robust by identifying issues and suggesting enhancements. This material was based on work supported in part by the U.S. Department of Energy, Office of Nuclear Energy, Nuclear Energy Advanced Modeling and Simulation (NEAMS) Program; by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research; and by the U.S. Department of Energy's Scientific Discovery through Advanced Computing program, under Contract DE-AC02-06CH11357.

## REFERENCES

1. Jain, R., and Tautges, T. J. RGG: Reactor Geometry (and mesh) Generator. *International Congress on the Advances in Nuclear Power Plants (ICAPP)*, Chicago (2012).
2. Tautges, T. J., and Jain, R. Creating geometry and mesh models for nuclear reactor core geometries using a lattice hierarchy-based approach. *Engineering with Computers*, *28*(4), 319–329 (2012).
3. SIGMA (Scalable Interfaces for Geometry and Mesh-Based Applications) website: http://sigma.mcs.anl.gov/
4. Tautges, T. J. CGM: A geometry interface for mesh generation, analysis and other applications. *Eng. Comput. 17*:486–490 (2005).
5. Tautges, T. J., Meyers, R., Merkley, K., Stimpson, C., and Ernst, C. MOAB: A mesh-oriented database, SAND2004-1592. Sandia National Laboratories, Albuquerque, NM (2004).
6. Sjaardema, G. D., Tautges, T. J, Wilson, T. J., Owen, S. J., Blacker, T. D., Bohnhoff, W. J., Edwards, T. L., Hipp, J. R., Lober, R. R., and Mitchell, S. A. CUBIT mesh generation environment, users manual, vol. 1. Sandia National Laboratories, Albuquerque (1994).
7. Jain, R., and Tautges, T. J. Generating unstructured nuclear reactor core meshes in parallel. 23rd *International Meshing Roundtable (IMR23),* 82, 351–363 (2014).
8. Computational Model Builder and RGG GUI tool, Kitware website: http://www.computationalmodelbuilder.org/rgg/
9. USNRC Technical Training Center, The Westinghouse PWR Technology manual, Rev 0195. Website: http://pbadupws.nrc.gov/docs/ML0230/ML023040131.pdf
10. Cho, N. Z. Benchmark problems in reactor and particle transport physics: Benchmark problem 3A. Website: http://nurapt.kaist.ac.kr/benchmark/ (2000).

## APPENDIX: Getting RGG GUI and MeshKit tools

The RGG GUI application can be obtained through two routes:

1. Download binaries for Windows/OSX/Linux: **http://www.computationalmodelbuilder.org/download/** .
2. Get the repository.

a. git clone **https://github.com/Kitware/RGG.git** .

b. Follow instructions to download dependencies, and configure/make/install.

CGM, MOAB, and MeshKit are maintained as open source software under an LGPL license. The MeshKit library uses several required and optional libraries, which must be built and installed prior to the MeshKit installation. All the SIGMA tools are currently supported on Linux and OSX, with partial support for Windows.

For MeshKit support, please contact **meshkit-dev@mcs.anl.gov**.